Seminar on "Equilibrium Computation"

Dagstuhl, 25-30 April 2010

Nucleolus Computation in Compact Coalitional Games

UNIVERSITÀ DELLA CALABRIA
CAMPUS DI ARCAVACATA

Gianluigi Greco, E. Malizia, L. Palopoli, and F. Scarcello

University of Calabria, Italy

"On the Complexity of Compact Coalitional Games", IJCAI'09

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\boxed{\mathcal{G} = \langle N, v \rangle,\ v : 2^N \mapsto \mathbb{R}}$$

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\mathcal{G} = \langle N, v \rangle, \; v : 2^N \mapsto \mathbb{R}$$

- Outcomes belong to the imputation set $X(\mathcal{G})$

$$x \in X(\mathcal{G}) \quad \left\{ \begin{array}{l} \bullet \; \textit{Efficiency} \\ \qquad x(N) = v(N) \\ \bullet \; \textit{Individual Rationality} \\ \qquad x_i \geq v(\{i\}), \quad \forall i \in N \end{array} \right.$$

The Model

- Players form *coalitions*
- Each coalition is associated with a *worth*
- A *total worth* has to be distributed

$$\mathcal{G} = \langle N, v \rangle, \; v : 2^N \mapsto \mathbb{R}$$

- **Solution Concepts** characterize outcomes in terms of
  - Fairness
  - Stability

Excess…

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

Excess...

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1,2\})) = v(\{1,3\}) = v(\{2,3\}) = 1$$
$$v(\{1,2,3\}) = 3$$

Excess…

- How fairness/stability can be measured?

$$e(S, x) = v(S) - x(S)$$

- The excess is a measure of the dissatisfaction of S

$x = (0, 0, 3) \longrightarrow e(\{1, 2\}, x) = v(\{1, 2\}) - (x_1 + x_2) = 1 - 0 = 1$

$x = (1, 2, 0) \longrightarrow e(\{1, 2\}, x) = v(\{1, 2\}) - (x_1 + x_2) = 1 - 3 = -2$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

...and the Nucleolus

- Arrange excess values in non-increasing order

$x = (1, 2, 0)$ $\theta(x) = (0, 0, -1, -1, -2, -2)$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

$x^* = (1, 1, 1)$ $\theta(x^*) = (-1, -1, -1, -1, -1, -1)$

$\boxed{x = (1, 2, 0) \quad \theta(x) = (0, 0, -1, -1, -2, -2)}$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1, 2\})) = v(\{1, 3\}) = v(\{2, 3\}) = 1$$
$$v(\{1, 2, 3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

$$\boxed{x^* = (1,1,1) \qquad \theta(x^*) = (-1,-1,-1,-1,-1,-1)}$$

$$x = (1,2,0) \qquad \theta(x) = (0,0,-1,-1,-2,-2)$$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1,2\})) = v(\{1,3\}) = v(\{2,3\}) = 1$$
$$v(\{1,2,3\}) = 3$$

...and the Nucleolus

- Arrange excess values in non-increasing order

Definition [Schmeidler]

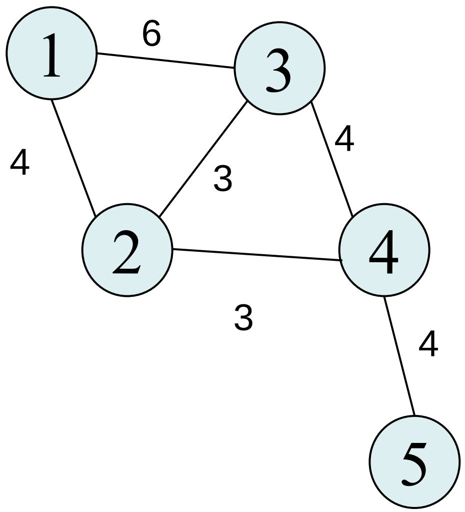
The *nucleolus* $\mathcal{N}(\mathcal{G})$ of a game $\mathcal{G}$ is the set
$\mathcal{N}(\mathcal{G}) = \{x \in X(\mathcal{G}) \mid \nexists y \in X(\mathcal{G}) \text{ s.t. } \theta(y) \prec \theta(x)\}$

$$\boxed{x^* = (1,1,1) \qquad \theta(x^*) = (-1,-1,-1,-1,-1,-1)}$$

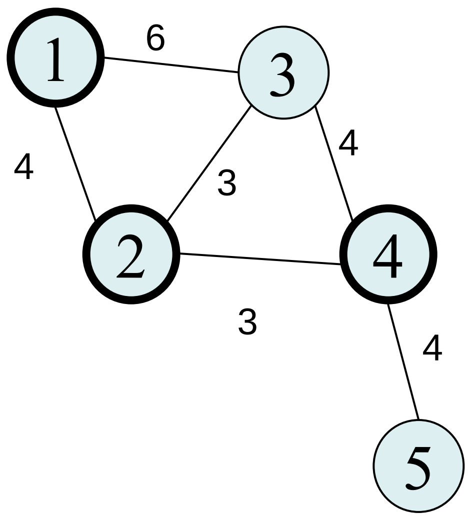
$$x = (1,2,0) \qquad \theta(x) = (0,0,-1,-1,-2,-2)$$

$$v(\{1\}) = v(\{2\}) = v(\{3\}) = 0$$
$$v(\{1,2\})) = v(\{1,3\}) = v(\{2,3\}) = 1$$
$$v(\{1,2,3\}) = 3$$

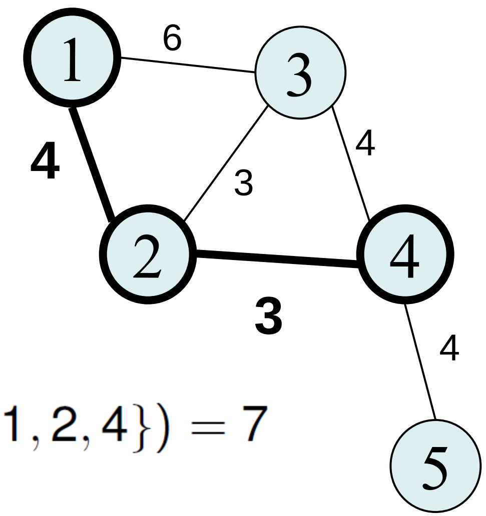
Compact Games



Compact Games

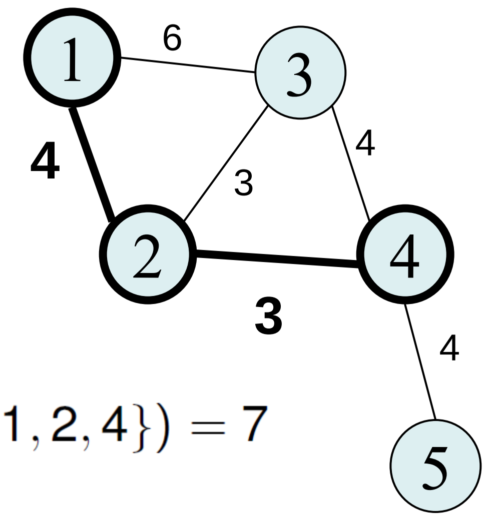


Compact Games



$v(\{1, 2, 4\}) = 7$

Compact Games

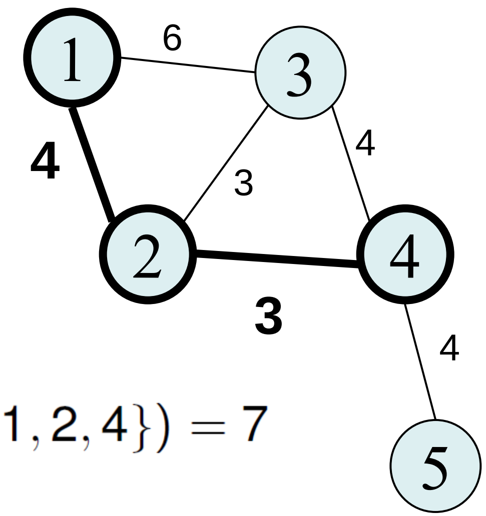


$v(\{1,2,4\}) = 7$

- *Graph Games* [Deng and Papadimitriou, 1994]
  - Computational issues of several solution concepts
  - The *(pre)nucleolus* can be computed in **P**

$$x_i^* = \frac{1}{2} \sum_{j \neq i} w_{i,j}$$

Compact Games



$v(\{1,2,4\}) = 7$

- *Graph Games* [Deng and Papadimitriou, 1994]
  - Computational issues of several solution concepts
  - The *(pre)nucleolus* can be computed in **P**

$$x_i^* = \frac{1}{2}\sum_{j \neq i} w_{i,j}$$

- *Cost allocation on trees* [Megiddo, 1978]
  - Polynomial time algorithm
- *Flow games* [Deng, Fang, and Sun, 2006]
  - Polynomial time algorithm on simple networks (unitary edge capacity)
  - **NP**-hard, in general
- *Weighted voting games* [Elkind and Pasechnik, 2009]
  - Pseudopolynomial algorithm

Computation Approaches

Succinct Linear Programs

Hardness Result

Further Solution Concepts

Computation Approaches

Succinct Linear Programs

Hardness Result

Further Solution Concepts

Kopelowitz, 1967

$$
\mathrm{LP}_1 \begin{cases} \min \epsilon_1 & \\ \quad e(S, x) \leq \epsilon_1 & \forall S \subset N, S \notin W_0 = \{\varnothing\} \\ \quad x \in X(\mathcal{G}) & \end{cases}
$$

Kopelowitz, 1967

$$\mathtt{LP}_1 \begin{cases} \min \epsilon_1 & \\ e(S,x) \leq \epsilon_1 & \forall S \subset N, S \notin W_0 = \{\emptyset\} \\ x \in X(\mathcal{G}) & \end{cases}$$

$$\mathtt{LP}_2 \begin{cases} \min \epsilon_2 & \\ e(S,x) = \epsilon_1^* & \forall S \in W_1 \\ e(S,x) \leq \epsilon_2 & \forall S \subset N, S \notin (W_0 \cup W_1) \\ x \in X(\mathcal{G}) & \end{cases}$$

where:

- $V_1 = \{x \mid (x, \epsilon_1^*)$ is an optimal solution to $\mathtt{LP}_1\}$
- $W_1 = \{S \subseteq N \mid e(S,x) = \epsilon_1^*,$ for every $x \in V_1\}$

Kopelowitz, 1967

$$\mathtt{LP}_k \left\{ \begin{array}{ll} \min \epsilon_k & \\ \quad e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\ldots,k-1\} \\ \quad e(S,x) \le \epsilon_k & \forall S \subset N, S \notin (W_0 \cup \cdots \cup W_{k-1}) \\ \quad x \in X(\mathcal{G}) & \end{array} \right.$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*)$ is an optimal solution to $\mathtt{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*,$ for every $x \in V_r\}$

How many iterations?

An Example Computation

$N = 1, \ldots, n, n+1, n+2$

$$v(N) = n+2$$
$$v(\{i\}) = 1,\ i \in \{1, \ldots, n\}$$
$$v(\{1, \ldots, n\}) = n$$
$$v(\{n+1\}) = v(\{n+2\}) = 0$$
$$v(\{n+1, n+2\}) = 2$$
$$v(S) = -\infty,\ |\{n+1, n+2\} \cap S| \geq 1,\ |\{1, \ldots, n\} \cap S| \geq 1,\ S \neq N$$

$S_1, S_2, \ldots \subset \{1, \ldots, n\}$ $|S_i| > 1$

$$v(S_i) = |S_i| - 1 + 2^{-i}$$

An Example Computation

$N = 1, ..., n, n+1, n+2$

$v(N) = n+2$
$v(\{i\}) = 1,\ i \in \{1, ..., n\}$
$v(\{1, ..., n\}) = n$
$v(\{n+1\}) = v(\{n+2\}) = 0$
$v(\{n+1, n+2\}) = 2$
$v(S) = -\infty,\ |\{n+1, n+2\} \cap S| \geq 1,$
$|\{1, ..., n\} \cap S| \geq 1,\ S \neq N$

$S_1, S_2, ... \subset \{1, ..., n\}\ \ |S_i| > 1$
$v(S_i) = |S_i| - 1 + 2^{-i}$

$\epsilon_1^* = 0$

$x^* = (1, ..., 1, x_{n+1}^*, x_{n+2}^*)$

$$\mathrm{LP}_1 \left\{ \begin{array}{l} \min \epsilon_1 \\ \quad n - x(\{1, ..., n\}) \leq \epsilon_1 \\ \quad 2 - x_{n+1} - x_{n+2} \leq \epsilon_1 \\ \quad x(\{1, ..., n\}) + x_{n+1} + x_{n+2} = n+2 \\ \quad x_i \geq 1,\ i \in \{1, ..., n\} \\ \quad \vdots \end{array} \right.$$

An Example Computation

$$N = 1, \ldots, n, n+1, n+2$$

$$v(N) = n+2$$
$$v(\{i\}) = 1,\ i \in \{1, \ldots, n\}$$
$$v(\{1, \ldots, n\}) = n$$
$$v(\{n+1\}) = v(\{n+2\}) = 0$$
$$v(\{n+1, n+2\}) = 2$$
$$v(S) = -\infty,\ |\{n+1, n+2\} \cap S| \geq 1,$$
$$|\{1, \ldots, n\} \cap S| \geq 1,\ S \neq N$$

$$S_1, S_2, \ldots \subset \{1, \ldots, n\} \quad |S_i| > 1$$
$$v(S_i) = |S_i| - 1 + 2^{-i}$$

$$\epsilon_1^* = 0$$

$$x^* = (1, \ldots, 1, x^*_{n+1}, x^*_{n+2})$$

The excess is constant

$$e(S_i, x^*) = v(S_i) - x^*(S_i) = -1 + 2^{-i}$$

An Example Computation

$$N = 1, ..., n, n+1, n+2$$

$$v(N) = n+2$$
$$v(\{i\}) = 1,\ i \in \{1, ..., n\}$$
$$v(\{1, ..., n\}) = n$$
$$v(\{n+1\}) = v(\{n+2\}) = 0$$
$$v(\{n+1, n+2\}) = 2$$
$$v(S) = -\infty,\ |\{n+1, n+2\} \cap S| \geq 1,\ |\{1, ..., n\} \cap S| \geq 1,\ S \neq N$$

$$S_1, S_2, ... \subset \{1, ..., n\} \quad |S_i| > 1$$
$$v(S_i) = |S_i| - 1 + 2^{-i}$$

$$\epsilon_1^* = 0$$
$$x^* = (1, \ldots, 1, x_{n+1}^*, x_{n+2}^*)$$

The excess is constant

$$e(S_i, x^*) = v(S_i) - x^*(S_i) = -1 + 2^{-i}$$

$$\begin{cases} e(S_i, x^*) \leq \epsilon_2 \\ \epsilon_2^* = -1 + 2^{-1} \end{cases}$$

An Example Computation

$$N = 1, \ldots, n, n+1, n+2$$

$$v(N) = n+2$$
$$v(\{i\}) = 1,\ i \in \{1, \ldots, n\}$$
$$v(\{1, \ldots, n\}) = n$$
$$v(\{n+1\}) = v(\{n+2\}) = 0$$
$$v(\{n+1, n+2\}) = 2$$
$$v(S) = -\infty,\ |\{n+1, n+2\} \cap S| \geq 1,\ |\{1, \ldots, n\} \cap S| \geq 1,\ S \neq N$$

$$S_1, S_2, \ldots \subset \{1, \ldots, n\} \quad |S_i| > 1$$
$$v(S_i) = |S_i| - 1 + 2^{-i}$$

$$\epsilon_1^* = 0$$

$$x^* = (1, \ldots, 1, x_{n+1}^*, x_{n+2}^*)$$

The excess is constant

$$e(S_i, x^*) = v(S_i) - x^*(S_i) = -1 + 2^{-i}$$

$$e(S_i, x^*) \leq \epsilon_3$$

$$\epsilon_2^* = -1 + 2^{-1} > \epsilon_3^* = -1 + 2^{-2}, \ldots >$$

Kopelowitz, 1967

$$
\mathrm{LP}_k \begin{cases} \min \epsilon_k \\ e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\ldots,k-1\} \\ e(S,x) \leq \epsilon_k & \forall S \subset N, S \notin (W_0 \cup \cdots \cup W_{k-1}) \\ x \in X(\mathcal{G}) \end{cases}
$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$

How many iterations?

Kopelowitz, 1967

$$\mathrm{LP}_k \begin{cases} \min \epsilon_k \\ \quad e(S,x) = \epsilon_r^* \qquad \forall S \in W_r, r \in \{1,\ldots,k-1\} \\ \quad e(S,x) \leq \epsilon_k \qquad \forall S \subset N, S \notin (W_0 \cup \cdots \cup W_{k-1}) \\ \quad x \in X(\mathcal{G}) \end{cases}$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$

Theorem

The algorithm performs $\Omega(2^n)$ steps, in some cases.

cf. Mashler, Peleg, and Shapley, 1979

$$
\mathtt{LP}_k \left\{
\begin{array}{ll}
\min \epsilon_k & \\
\quad e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\ldots,k-1\} \\
\quad e(S,x) \leq \epsilon_k & \forall S \subset N, S \notin (W_0 \cup \cdots \cup W_{k-1}) \\
\quad x \in X(\mathcal{G}) &
\end{array}
\right.
$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathtt{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$

cf. Mashler, Peleg, and Shapley, 1979

$$\mathrm{LP}_k \begin{cases} \min \epsilon_k \\ \quad e(S,x) = \epsilon_r^* \qquad \forall S \in W_r, r \in \{1,\dots,k-1\} \\ \quad e(S,x) \leq \epsilon_k \qquad \forall S \subset N, S \notin \cancel{(W_0 \cup \dots \cup W_{k-1})} \\ \quad x \in X(\mathcal{G}) \end{cases}$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*)$ is an optimal solution to $\mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*,$ for every $x \in V_r\}$

$$\boxed{\{S \subseteq N \mid x(S) = y(S), \forall x, y \in V_{k-1}\}}$$

cf. Mashler, Peleg, and Shapley, 1979

$$
\mathrm{LP}_k
\left\{
\begin{array}{ll}
\min \epsilon_k & \\
\quad e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\dots,k-1\} \\
\quad e(S,x) \leq \epsilon_k & \forall S \subset N, S \notin \cancel{(W_0 \cup \dots \cup W_{k-1})} \\
\quad x \in X(\mathcal{G}) &
\end{array}
\right.
$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*)$ is an optimal solution to $\mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*,$ for every $x \in V_r\}$

$$\boxed{\{S \subseteq N \mid x(S) = y(S), \forall x, y \in V_{k-1}\}}$$

[Kern and Paulusuma, 2003]

LP Approaches over Compact Games

$$\mathrm{LP}_k \begin{cases} \min \epsilon_k \\ e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\ldots,k-1\} \\ e(S,x) \leq \epsilon_k & \forall S \subset N, S \notin \mathcal{F}_{k-1} \\ x \in X(\mathcal{G}) \end{cases}$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$
- $\mathcal{F}_{k-1} = \{S \subseteq N \mid x(S) = y(S), \forall x, y \in V_{k-1}\}$

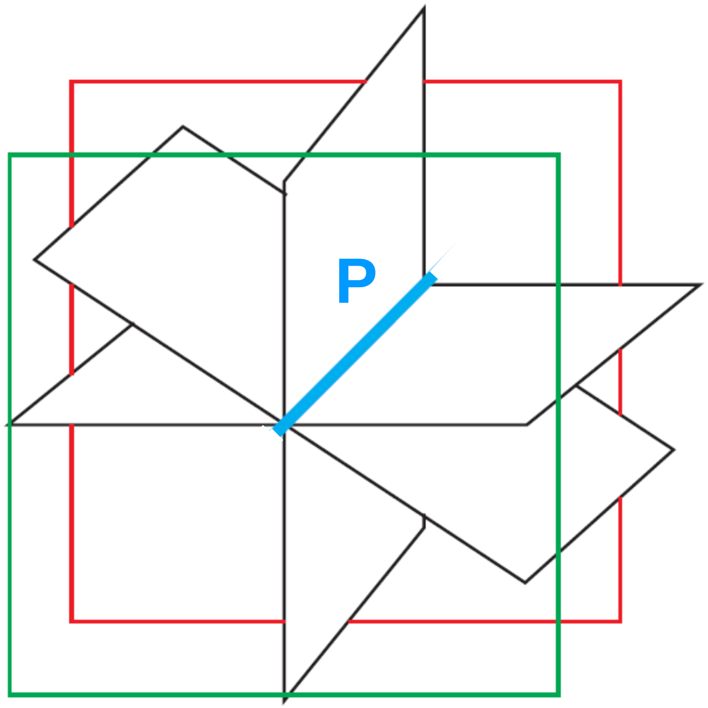
- In compact games, two problems have to be faced:
  - **(P1)** Sets W and $\mathcal{F}$ contain exponentially many elements, but we would like to avoid listing them explicitly
  - **(P2)** Translate LP (complexity) results to “succinct programs”

(P1): A Convenient Representation

$W \qquad \mathcal{F}$

fixed inequalities

equalities + implied equalities

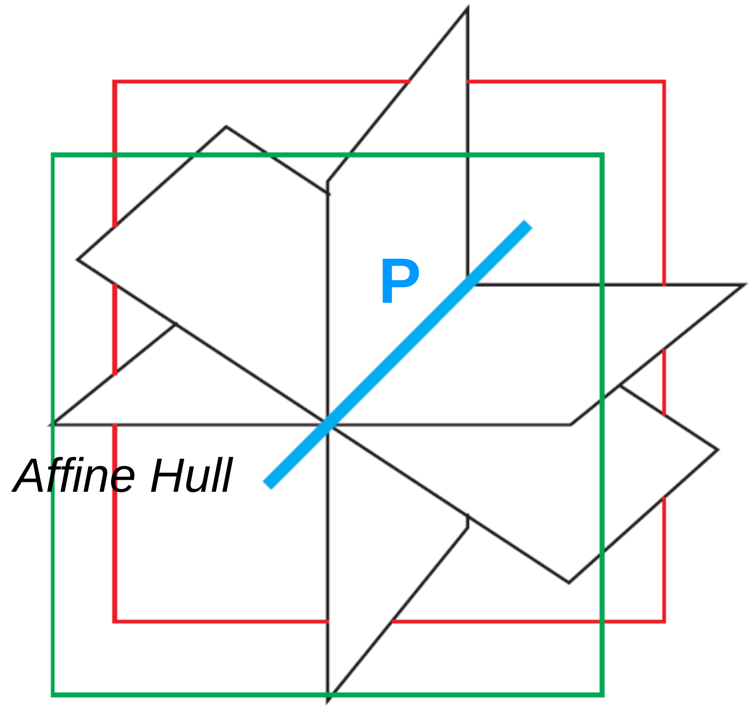


(P1): A Convenient Representation

W $\mathcal{F}$

fixed inequalities

equalities + implied equalities



Theorem

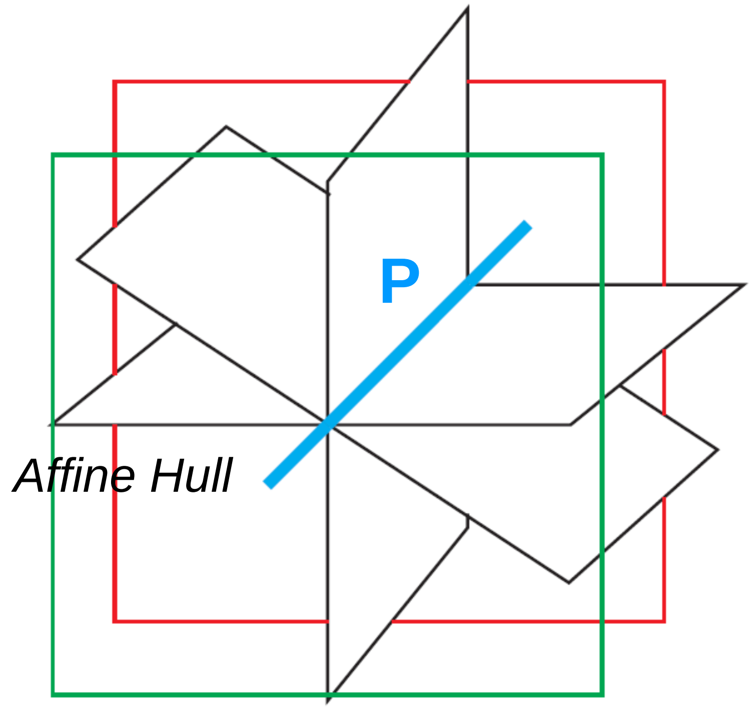
- $aff.hull(V_k) = \text{solutions for equalities over } W_k \cup W_{k-1} \cup \cdots \cup W_1$

(P1): A Convenient Representation

$W \quad \mathcal{F}$

fixed inequalities

equalities + implied equalities

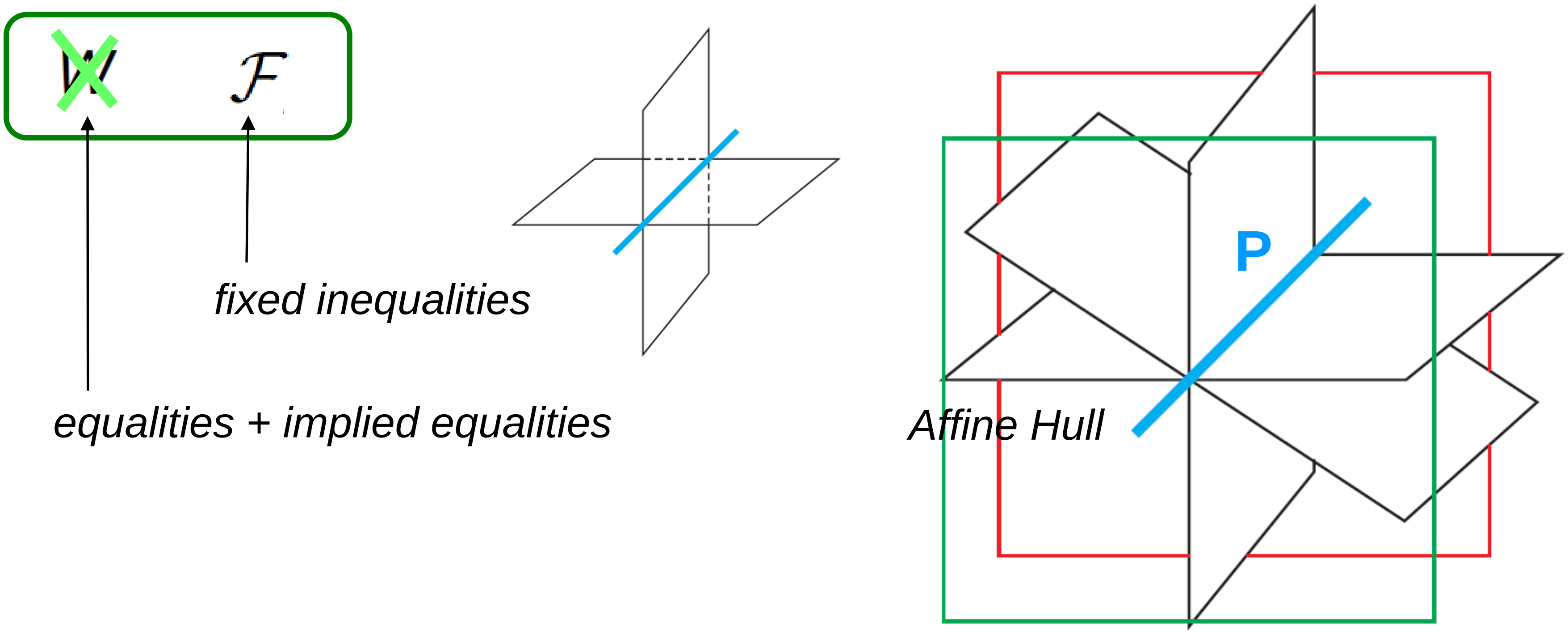


Theorem

- $aff.hull(V_k) = \text{solutions for equalities over } \underset{\substack{\uparrow \\ \text{Implied equalities}}}{W_k} \cup \underbrace{W_{k-1} \cup \cdots \cup W_1}_{\text{equalities}}$

  $\{S \subseteq N \mid e(S, x) = \epsilon_k^*, \text{ for every } x \in V_k\}$

(P1): A Convenient Representation



Theorem

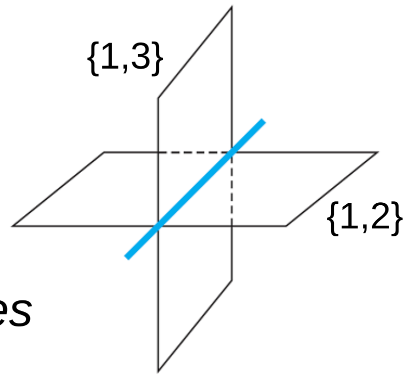
- $aff.hull(V_k) =$ *solutions for equalities over* $W_k \cup W_{k-1} \cup \cdots \cup W_1$
- *A basis* $\mathcal{B}_k$ *for* $aff.hull(V_k)$ *contains n vectors at most*

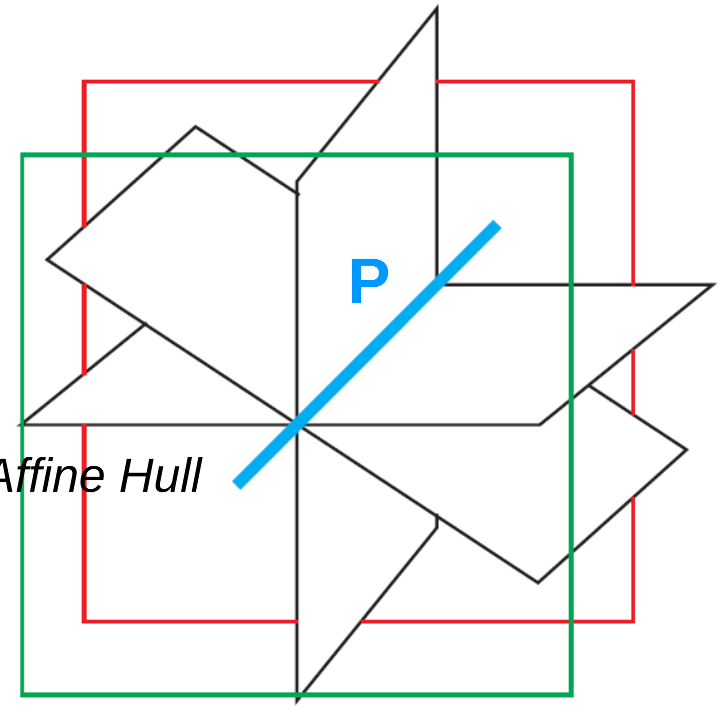
(P1): A Convenient Representation

$\cancel{W}$ $\mathcal{F}$

fixed inequalities

equalities + implied equalities

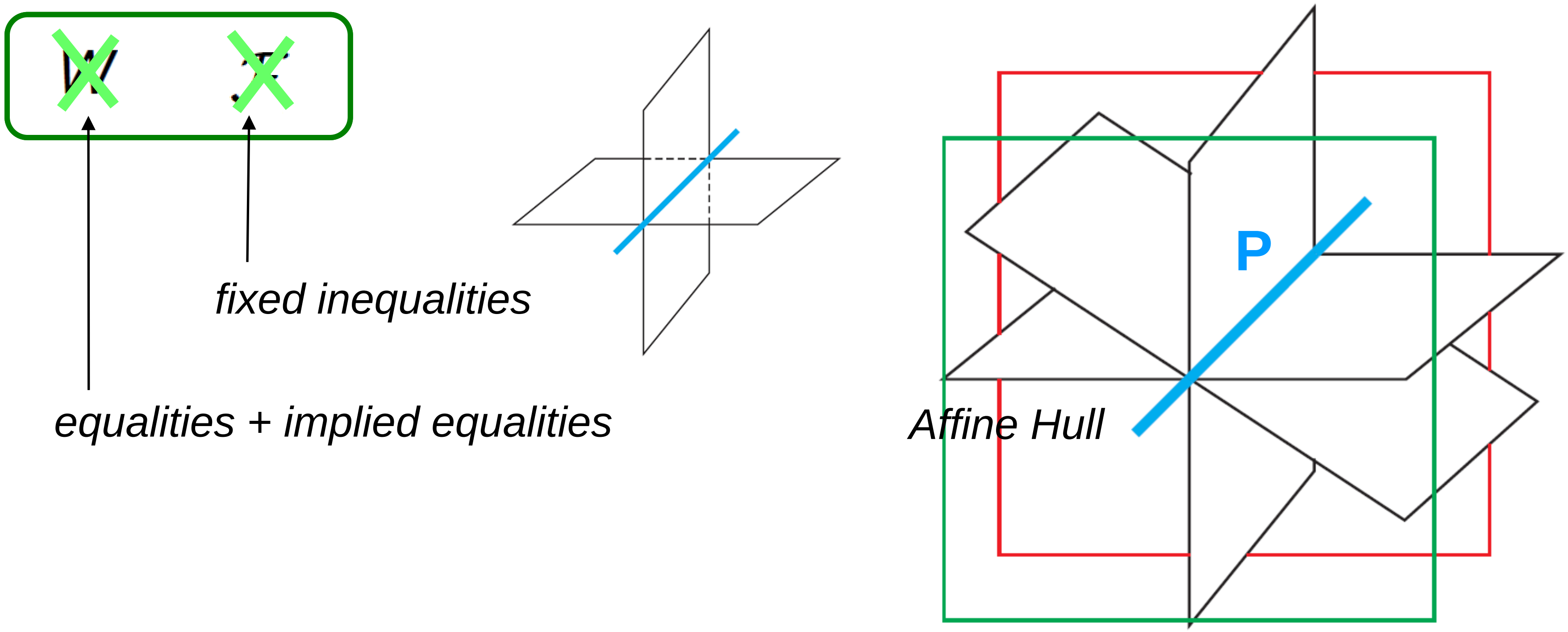




Theorem

- *aff.hull*(V_k) = *solutions for equalities over* $W_k \cup W_{k-1} \cup \cdots \cup W_1$
- *A basis* $\mathcal{B}_k$ *for aff.hull*(V_k) *contains n vectors at most*
- $S \in \mathcal{F}_k$ *iff S is a linear combination of the* indicator *vectors for* $\mathcal{B}_k$

(P1): A Convenient Representation



Theorem

- $aff.hull(V_k) =$ *solutions for equalities over* $W_k \cup W_{k-1} \cup \cdots \cup W_1$
- *A basis* $\mathcal{B}_k$ *for* $aff.hull(V_k)$ *contains n vectors at most*
- $S \in \mathcal{F}_k$ *iff S is a linear combination of the* indicator *vectors for* $\mathcal{B}_k$

(P1): A Convenient Representation

$$v : 2^N \mapsto \mathbb{R}$$

$$+$$

$$aff.hull \quad \Longrightarrow \quad \mathrm{LP}_{k+1}$$

$$\xrightarrow{\;i\;} \boxed{\mathrm{LP}} \xrightarrow{\;\textit{i-th inequality}\;}$$

Theorem

- $aff.hull(V_k)$ = *solutions for equalities over* $W_k \cup W_{k-1} \cup \cdots \cup W_1$
- *A basis* $\mathcal{B}_k$ *for* $aff.hull(V_k)$ *contains* n *vectors at most*
- $S \in \mathcal{F}_k$ *iff* S *is a linear combination of the* indicator *vectors for* $\mathcal{B}_k$

Computation Approaches

Succinct Linear Programs

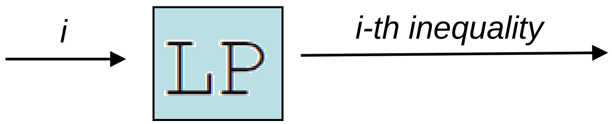
Hardness Result

Further Solution Concepts

(P2) Computation Problems

- In compact games, two problems have to be faced:
  - **(P1)** Sets W and $\mathcal{F}$ contain exponentially many elements, but we would like to avoid listing them explicitly
  - **(P2)** Translate LP (complexity) results to "succinct programs"

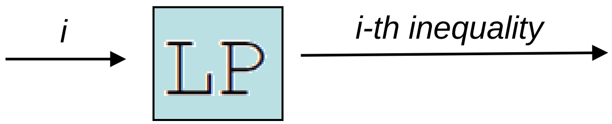
(P2) Computation Problems



Problem	Result
Membership	in co-**NP**
NonEmptiness	in co-**NP**
Dimension	in **NP**
AffineHullComputation	in $\mathbf{F\Delta_2^P}$
OptimalValueComputation	in $\mathbf{F\Delta_2^P}$
FeasibleVectorComputation	in $\mathbf{F\Delta_2^P}$
OptimalVectorComputation	in $\mathbf{F\Delta_2^P}$

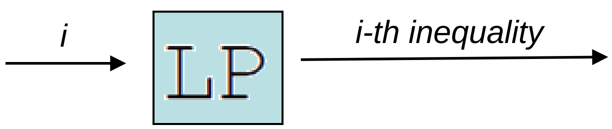
- In compact games, two problems have to be faced:
  - **(P1)** Sets W and $\mathcal{F}$ contain exponentially many elements, but we would like to avoid listing them explicitly
  - **(P2)** Translate LP (complexity) results to "succinct programs"

Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

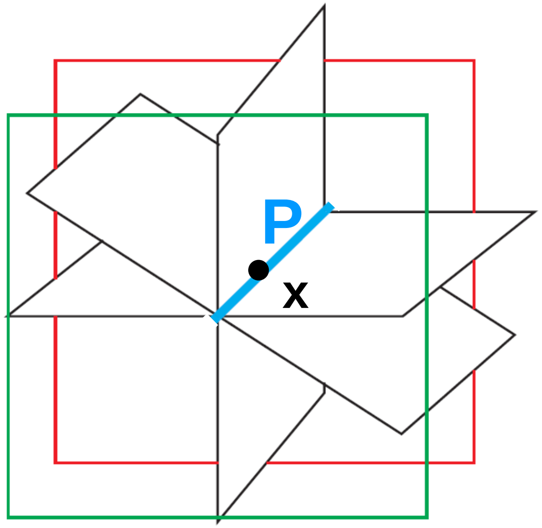
Complexity Results



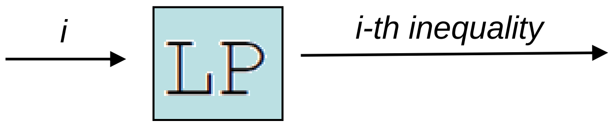
Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Trivial

- Given a vector **x**, we can:
  - Guess an index *i*
  - Check that the *i-th inequality* is not satisfied by **x**

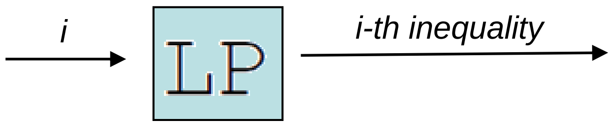


Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Complexity Results

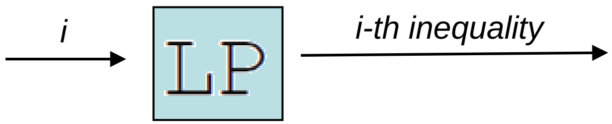


Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Proof

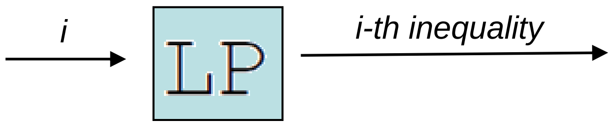
- By **Helly's theorem**, we can solve the complementary problem in **NP**:
  - Guess **n+1** inequalities
  - Check that they are not satisfiable (in polynomial time)

Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Proof Overview

(1) The dimension is **n-k** at most, if there are at least **k** linear independent implied equalities

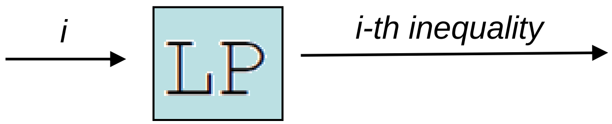
(2) In order to check that the *i-th* inequality is an implied one,

we can guess in **NP** a **support set** *W(i),* again by Helly's theorem:

- **n** inequalities + the *i-th* inequality treated as strict
- *W(i)* is not satisfiable, which can be checked in polynomial time

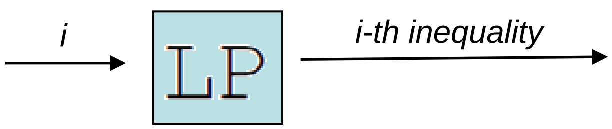
- Guess **k** implied equalities plus their support sets
- Check that they are linear independent

Complexity Results



Problem	**Result**
Membership	in co-**NP**
NonEmptiness	in co-**NP**
Dimension	in **NP**
AffineHullComputation	in $\mathbf{F\Delta}_2^P$
OptimalValueComputation	in $\mathbf{F\Delta}_2^P$
FeasibleVectorComputation	in $\mathbf{F\Delta}_2^P$
OptimalVectorComputation	in $\mathbf{F\Delta}_2^P$

Complexity Results

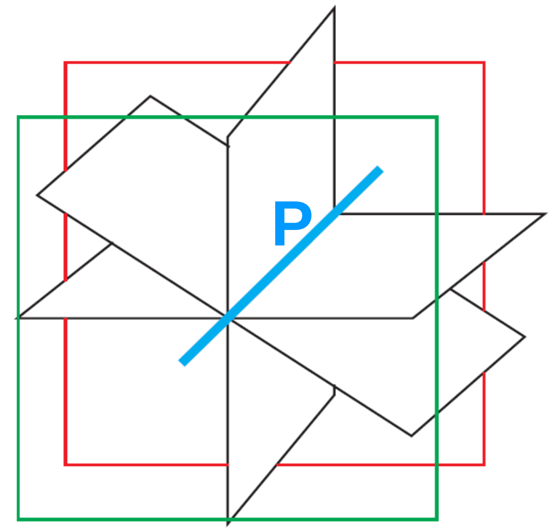


Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

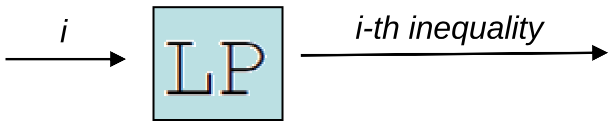
Proof

(1) Compute the dimension **n-k**, with a *binary search* invoking an **NP** oracle

(2) Guess **k** implied equalities plus their support sets

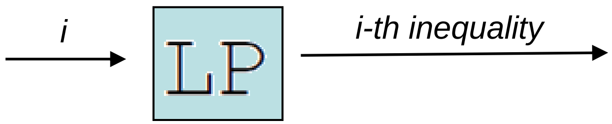


Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta}_2^P$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta}_2^P$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta}_2^P$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta}_2^P$

Routine

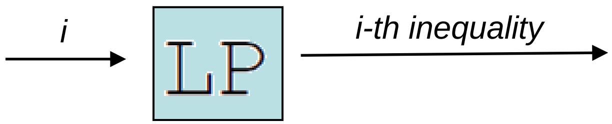
(1) Bfs can be represented with polynomially many bits

(2) LP induces a polytope and hence the optimum is achieved on some bfs.

(3) Perform a *binary search* over the range of the optimum solution:

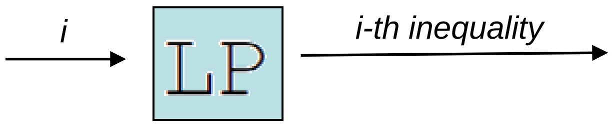
- Add the current value as a constraint, and check satisfiability

Complexity Results



Problem	**Result**
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Complexity Results

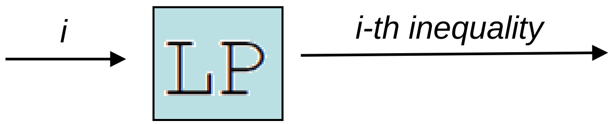


Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Routine

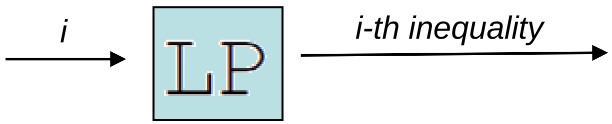
- LP induces a polytope
- Compute the lexicographically maximum bfs solution, by iterating over the various components, and treating each of them as an objective function to be optimized.

Complexity Results



Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

Complexity Results



Problem	Result
MembershiP	in co-**NP**
NonEmptiness	in co-**NP**
Dimension	in **NP**
AffineHullComputation	in $\mathbf{F\Delta}_2^P$
OptimalValueComputation	in $\mathbf{F\Delta}_2^P$
FeasibleVectorComputation	in $\mathbf{F\Delta}_2^P$
OptimalVectorComputation	in $\mathbf{F\Delta}_2^P$

Routine

(1) Compute the optimum value

(2) Define LP' as LP plus the constraint stating that the objective function must equal the optimum value

(3) Compute a feasible value for LP'

Putting It All Togheter

$$\mathrm{LP}_k \begin{cases} \min \epsilon_k \\ e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\dots,k-1\} \\ e(S,x) \le \epsilon_k & \forall S \subset N, S \notin \mathcal{F}_{k-1} \\ x \in X(\mathcal{G}) \end{cases}$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$
- $\mathcal{F}_{k-1} = \{S \subseteq N \mid x(S) = y(S), \forall x, y \in V_{k-1}\}$

$v : 2^N \mapsto \mathbb{R}$ + *aff.hull* $\longrightarrow \mathrm{LP}_{k+1}$

Problem	Result
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

M.P.S. — **Compact Encoding** — **Algorithms in $\mathrm{F}\Delta_2^P$**

- In compact games, two problems have to be faced:
  - **(P1)** Sets W and $\mathcal{F}$ contain exponentially many elements, but we would like to avoid listing them explicitly
  - **(P2)** Translate LP (complexity) results to “succinct programs”

Putting It All Togheter

$$\mathrm{LP}_k \begin{cases} \min \epsilon_k & \\ e(S,x) = \epsilon_r^* & \forall S \in W_r, r \in \{1,\dots,k-1\} \\ e(S,x) \le \epsilon_k & \forall S \subset N, S \notin \mathcal{F}_{k-1} \\ x \in X(\mathcal{G}) & \end{cases}$$

where:

- $V_r = \{x \mid (x, \epsilon_r^*) \text{ is an optimal solution to } \mathrm{LP}_r\}$
- $W_r = \{S \subseteq N \mid e(S,x) = \epsilon_r^*, \text{ for every } x \in V_r\}$
- $\mathcal{F}_{k-1} = \{S \subseteq N \mid x(S) = y(S), \forall x, y \in V_{k-1}\}$

$v : 2^N \mapsto \mathbb{R}$

$+$

aff.hull $\longrightarrow \mathrm{LP}_{k+1}$

Problem	**Result**
MEMBERSHIP	in co-**NP**
NONEMPTINESS	in co-**NP**
DIMENSION	in **NP**
AFFINEHULLCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVALUECOMPUTATION	in $\mathbf{F\Delta_2^P}$
FEASIBLEVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$
OPTIMALVECTORCOMPUTATION	in $\mathbf{F\Delta_2^P}$

M.P.S. | **Compact Encoding** | **Algorithms in $\mathbf{F\Delta_2^P}$**

Theorem

Computing the nucleolus is feasible in $\mathbf{F\Delta_2^P}$. Thus, deciding whether an imputation is the nucleolus is feasible in $\mathbf{\Delta_2^P}$.

Computation Approaches

Succinct Linear Programs

Hardness Result

Further Solution Concepts

Checking Problem

Theorem

Deciding whether an imputation is the nucleolus is $\mathbf{\Delta_2^P}$-hard. Thus, it is $\mathbf{\Delta_2^P}$-complete.

Checking Problem

Theorem

Deciding whether an imputation is the nucleolus is $\mathbf{\Delta_2^P}$-hard. Thus, it is $\mathbf{\Delta_2^P}$-complete.

Proof (Reduction for Graph Games: *The cost of individual rationality!*)

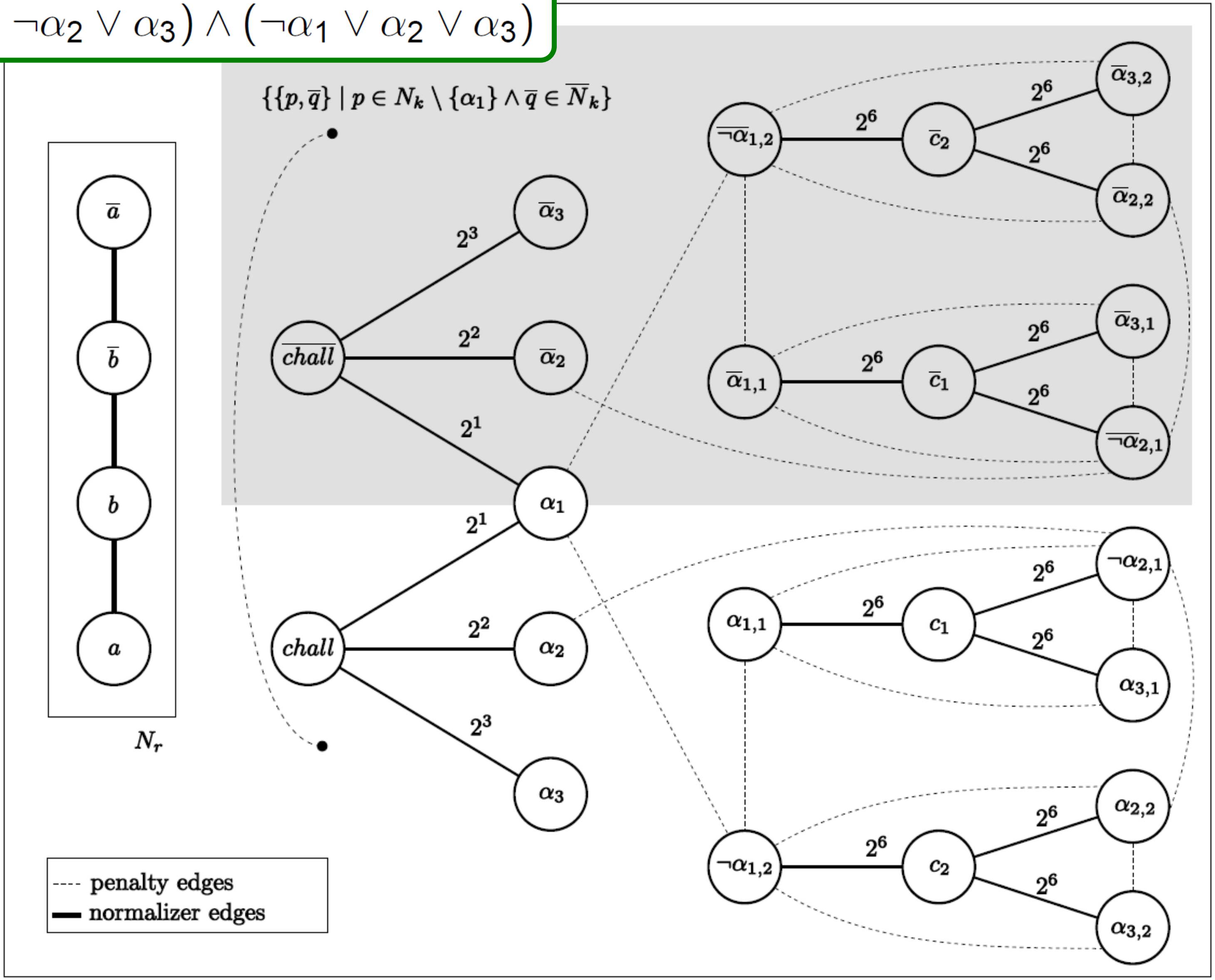
- Deciding the truth value of the least significant variable in the lexicographically maximum satisfying assignment

$$\boxed{\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)}$$

$\alpha_1 < \alpha_2 < \alpha_3$

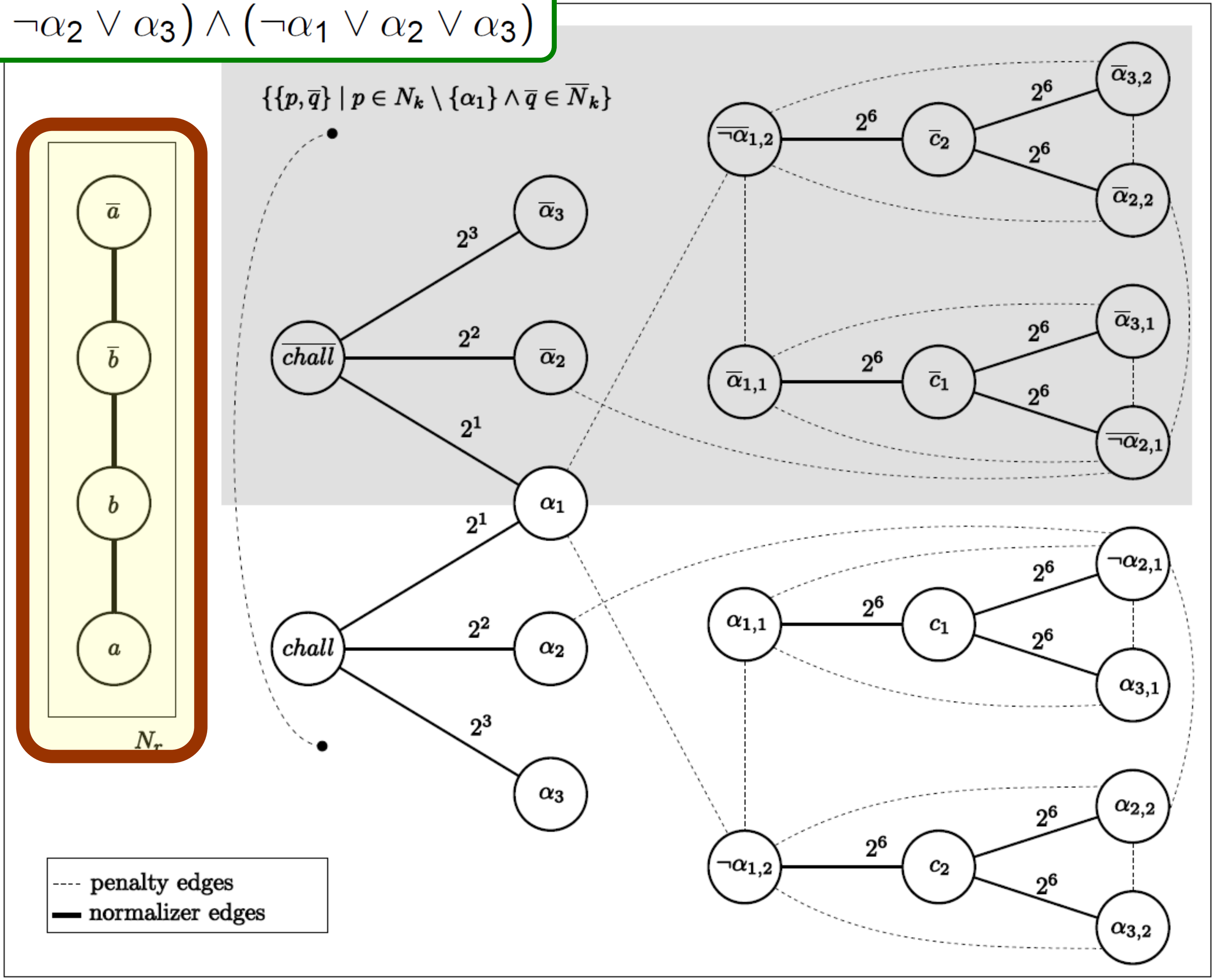
Overview of the Reduction

$$\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$$



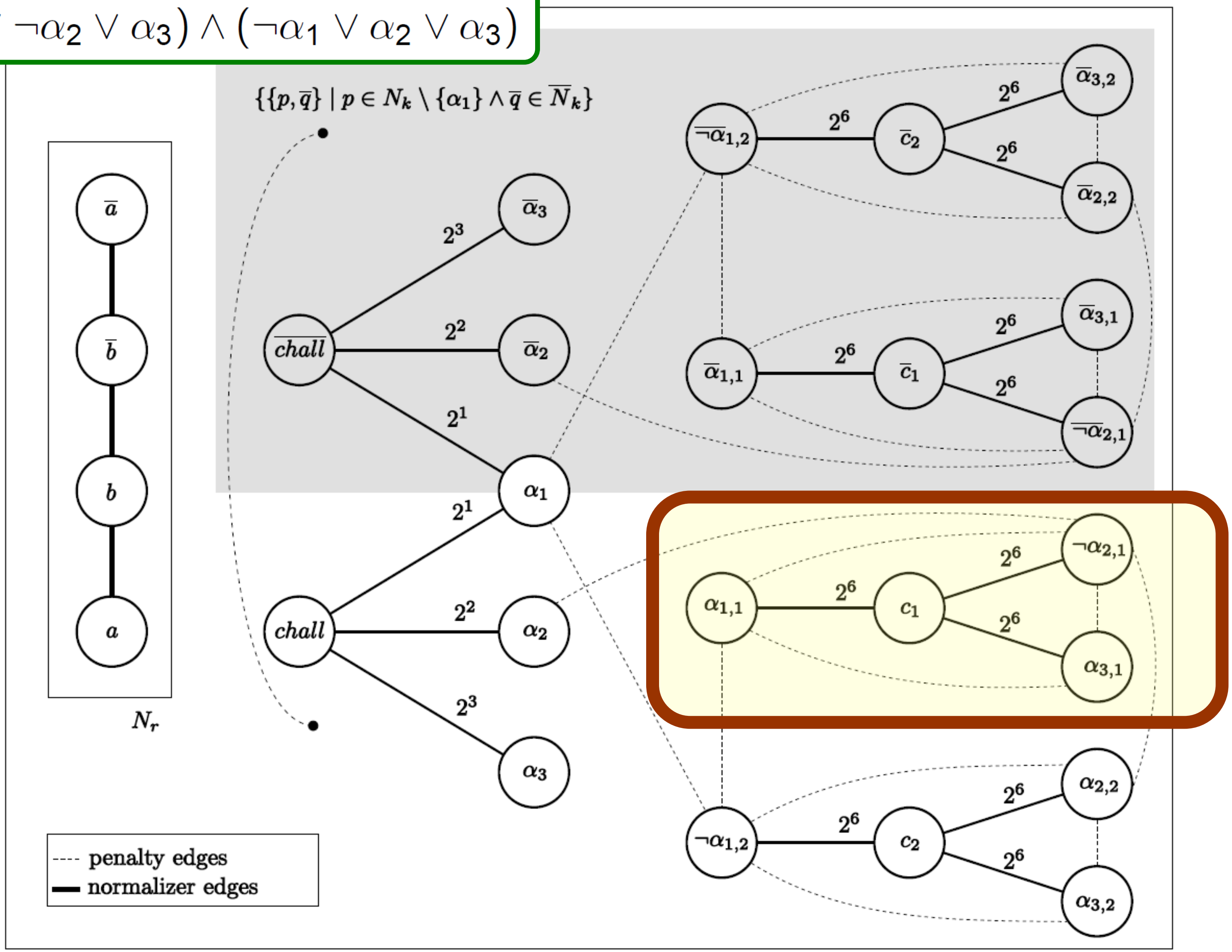
Overview of the Reduction

$$\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$$



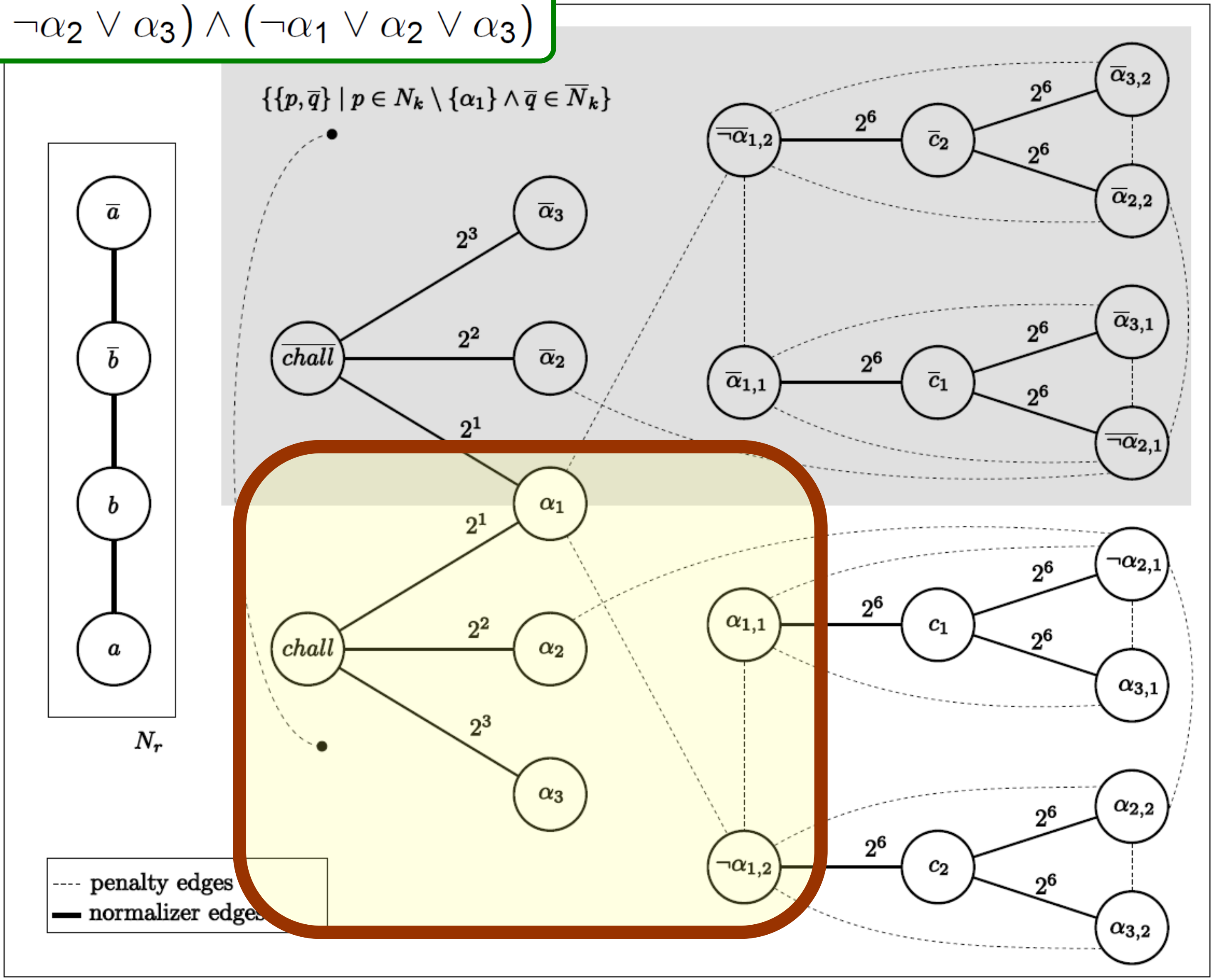
Overview of the Reduction

$$\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$$



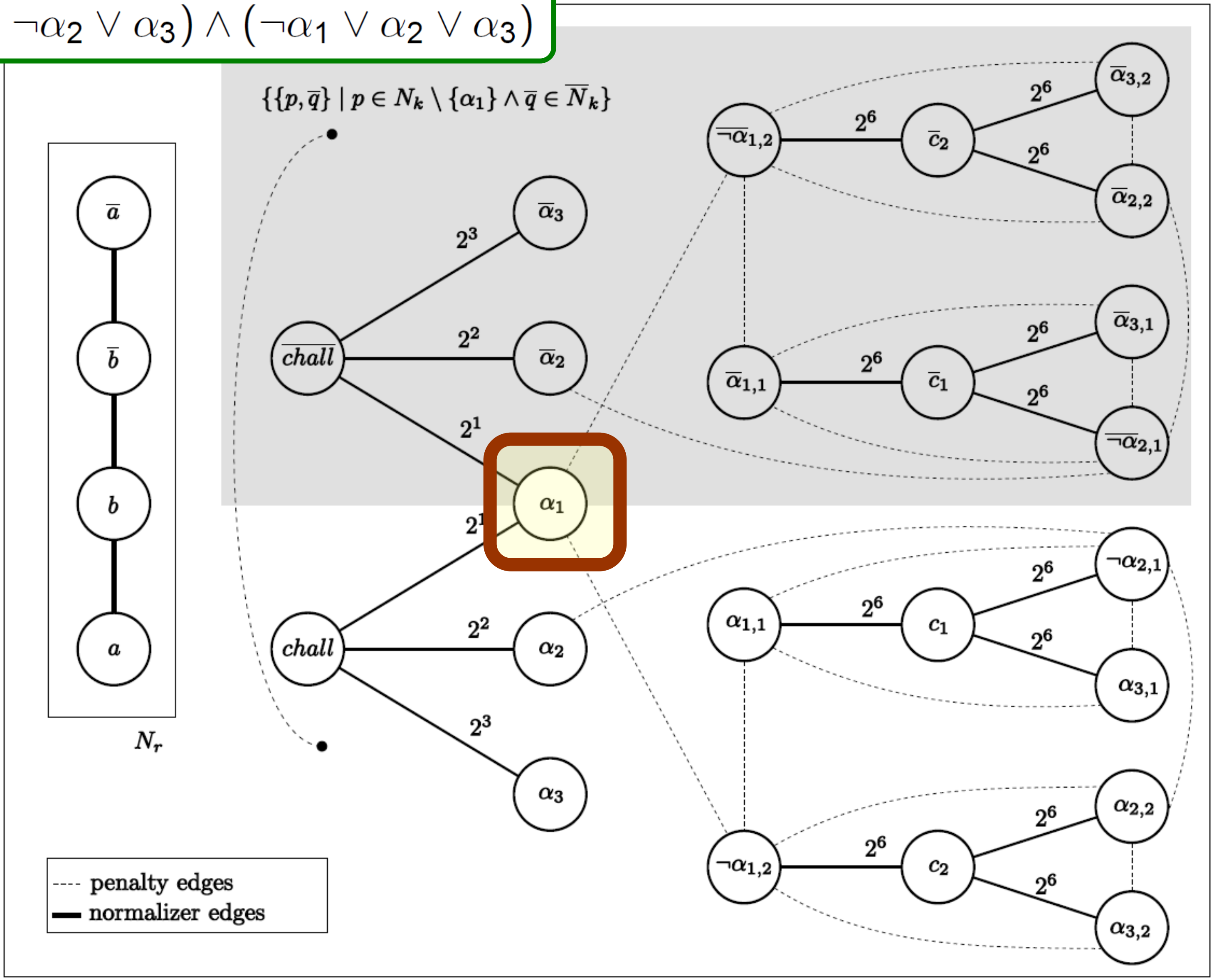
Overview of the Reduction

$$\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$$



Overview of the Reduction

$$\hat{\phi} = (\alpha_1 \vee \neg\alpha_2 \vee \alpha_3) \wedge (\neg\alpha_1 \vee \alpha_2 \vee \alpha_3)$$



Computation Approaches

Succinct Linear Programs

Hardness Result

Further Solution Concepts

- Core
- Kernel
- Bargaining Set

Computation Approaches

Succinct Linear Programs

Hardness Result

Further Solution Concepts

Core

Kernel

Bargaining Set

Stable Sets

Thank you!